

# Multiple Error-Correcting WOM-Codes

Eitan Yaakobi, Paul H. Siegel, Alexander Vardy, and Jack K. Wolf  
 University of California, San Diego La Jolla, CA 92093 – 0401, USA  
 Emails: {eyaakobi, psiegel, avardy, jwolf}@ucsd.edu

**Abstract**—A *Write Once Memory (WOM)* is a storage medium with binary memory elements, called *cells*, that can change from the zero state to the one state only once. Examples of WOMs are punch cards, optical disks, and more recently flash memories. WOM-codes were first presented by Rivest and Shamir and are designed for efficiently storing and updating data in the WOM. A  $\mathcal{WC}[n, k, t]$  *WOM-Code*  $\mathcal{C}_W$  is a coding scheme for storing  $k$  information bits in  $n$  cells  $t$  times. At each write, the state of each cell can be changed, provided that the cell is changed from the zero state to the one state. The *WOM-Rate* of  $\mathcal{C}_W$ , defined to be  $\mathcal{R}^t(\mathcal{C}_W) = kt/n$ , indicates the total amount of information that is possible to store in a cell in  $t$  writes. Two WOM-code constructions that can correct a single cell-error were presented by Zémor and Cohen. In this paper, we present another construction of a single-error-correcting WOM-codes with a better WOM-rate. Our construction can be adjusted also for single-error-detection, double-error-correction, and triple-error-correction. For the latter case, we use triple-error-correcting BCH-like codes, which were showed by Kasami and more recently described again by Bracken and Helleseeth.

## I. INTRODUCTION

Write Once Memory (WOM) codes were first presented by Rivest and Shamir almost three decades ago [13]. The codes were designed for memories which consist of binary memory elements that can only be changed from a zero state to a one state. Examples of such memories are punch cards and optical disks. Since then, more research results have appeared on this topic [2], [4], [5], [14]. Recently, such codes have been suggested for application to flash memories [9], [10], [12].

The atomic memory element in flash memories is a floating gate cell. The cell is electrically charged with electrons and can have multiple levels corresponding to a different numbers of electrons in the cell [6]. Here, we are concerned with cells that take on two levels. A group of cells, typically  $2^{20}$  cells, constitutes of a block. While it is possible to increase an individual cell level in the block, it is impossible to reduce its level, unless the entire block is erased and then reprogrammed [6]. This model is a generalization of the WOM model [10], [12]. In fact, it was already described before in [4], [5], however without mentioning the connection to flash memories.

In the WOM model, the problem that has received the most attention is: *what is the minimum number of cells  $n$  required to store  $k$  bits  $t$  times?* Or, alternatively: *what is the maximum number of bits  $k$  that can be written  $t$  times using  $n$  cells?* A code that is designed for this problem is called a **WOM-Code**  $\mathcal{C}_W$ . The **WOM-rate**  $\mathcal{R}^t(\mathcal{C}_W)$  of a WOM-code  $\mathcal{C}_W$  with  $t$  writes is the ratio of the total number of bits written to the memory,  $kt$ , to the number of cells  $n$ ,  $\mathcal{R}^t(\mathcal{C}_W) = \frac{kt}{n}$ .

The first example of a WOM-Code was presented by Rivest and Shamir for storing two bits twice using only three cells [13]. Since then, several more WOM-code constructions were presented, including tabular WOM-codes and “linear” WOM-codes [13]. In [2] and [7], a “coset-coding” technique based upon binary linear codes is used to construct WOM-codes. The WOM model has been generalized for the multi-level case in [5] and was later discussed again in [4].

Even though the problem of adapting WOM-codes to handle memory errors was suggested in the first WOM-codes paper [13], the first construction of codes addressing this problem appeared a few years later by Zémor [15] and Zémor and Cohen [14]. The capacity of a noisy WOM was studied by Heegard [8]. Recently, in [9], Jiang discussed the generalization of error-correcting WOM-codes for the flash/floating codes model [10], [12].

Two constructions of error-correcting WOM-codes were given in [14]. Both constructions correct a single cell-error during the writes. The first construction, based on a double-error-correcting BCH code, enables one to write  $k$  bits using  $n = 2^k - 1$  cells  $t \approx n/15.42$  times, so its WOM-rate is roughly  $\frac{k}{15.42} \approx \frac{\log_2(15.42t+1)}{15.42}$ . The second construction, which uses the same number of cells, is based on a triple-error-correcting BCH code and stores  $2k$  bits  $t \approx n/26.9$  times. Its WOM-rate is approximately  $\frac{2k}{26.9} \approx \frac{\log_2(26.9t+1)}{13.45}$ . While there are different ways to compare WOM-codes, we find that the appropriate figure of merit is to compare the WOM-rates under the assumption of a fixed number of writes. In general, the more writes the WOM-code can support, the better the rate it can achieve. The second construction in [14] is superior to the first one as it achieves a better WOM-rate even though its number of writes is smaller.

A trivial scheme to construct an  $e$ -error-correcting WOM-code is based upon an existing WOM-code  $\mathcal{C}_W$  that stores  $k$  bits  $t$  times in  $n$  cells. In this scheme, each one of the  $n$  cells is replicated  $2e + 1$  times so it is possible to correct any  $e$  or fewer cell-errors. If the WOM-code  $\mathcal{C}_W$  has WOM-rate  $\mathcal{R}^t(\mathcal{C}_W) = \frac{kt}{n}$ , then the generated  $e$ -error-correcting WOM-code has WOM-rate  $\frac{kt}{(2e+1)n} = \frac{1}{2e+1} \mathcal{R}^t(\mathcal{C}_W)$ . For example, in [7], a WOM-code which stores  $k$  bits  $5 \cdot 2^{k-4} + 1$  times using  $2^k - 1$  cells, for  $k \geq 4$  is presented. If we use this WOM-code to construct a single-error-correcting WOM-code, then its rate,  $\frac{1}{3} \frac{k(5 \cdot 2^{k-4} + 1)}{2^k - 1} \approx \frac{\log_2(3.2(t-1))}{9.6}$ , outperforms for  $t$  large enough the WOM-rate of the two constructions in [14].

## II. PRELIMINARIES

In this work, the memory elements, called *cells*, have two states: zero and one. At the beginning, all the cells are in their zero state. A **programming operation** of a cell is changing its state from zero to one. This operation is irreversible in the sense that a cell cannot change its state from one to zero unless the entire memory is erased. The **memory-state vectors** are all the binary vectors of length  $n$ ,  $\{0, 1\}^n$ . The **data vectors** are the set of all binary vectors of length  $k$ ,  $\{0, 1\}^k$ . Any WOM-code  $\mathcal{C}_W$  is specified by its encoding map  $\mathcal{E}_{\mathcal{C}_W}$  and decoding map  $\mathcal{D}_{\mathcal{C}_W}$ . The **decoding map**  $\mathcal{D}_{\mathcal{C}_W} : \{0, 1\}^n \rightarrow \{0, 1\}^k$  assigns to each memory-state vector  $c \in \{0, 1\}^n$  its corresponding data vector  $v = \mathcal{D}_{\mathcal{C}_W}(c) \in \{0, 1\}^k$ . The **encoding map**  $\mathcal{E}_{\mathcal{C}_W} : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \cup \{E\}$  indicates for each new data vector  $v \in \{0, 1\}^k$  and memory-state vector  $c \in \{0, 1\}^n$ , a new memory-state vector  $c' = \mathcal{E}_{\mathcal{C}_W}(v, c)$

such that  $\mathcal{D}_{C_W}(c') = v$ , and  $c_i \leq c'_i$ , for all  $1 \leq i \leq n$ . In case there such a  $c' \in \{0, 1\}^n$  does not exist, the value of the encoding map is  $\mathcal{E}_{C_W}(v, c) = E$ .

**Definition.** A  $\mathcal{WC}[n, k, t]$  **WOM-Code**  $C_W(\mathcal{E}_{C_W}, \mathcal{D}_{C_W})$  is a coding scheme which consists of  $n$  cells and is defined by its encoding and decoding maps, denoted by  $\mathcal{E}_{C_W}$  and  $\mathcal{D}_{C_W}$ , respectively. The WOM-code  $C_W$  guarantees any  $t$  writes of a  $k$ -bit data vector  $v$  without producing the block erasure symbol  $E$ . The **WOM-Rate** of the code  $C_W$  is defined as  $\mathcal{R}^t(C_W) = \frac{kt}{n}$ .

**Remark 1.** It is possible to generalize the definition of WOM-Codes to allow an arbitrary number of bits or symbols to be stored at each write. In this work we focus only on the case where the *same number of bits* is written at each write.

The following definitions are also used in our work:

- 1) A  $\mathcal{WC}[n, k, t]$  WOM-code that can correct  $e$  errors is called a  $\mathcal{WC}[n, k, t]$  **e-Error-Correcting WOM-Code**.
- 2) A  $\mathcal{WC}[n, k, t]$  WOM-code that can detect  $e$  errors is called a  $\mathcal{WC}[n, k, t]$  **e-Error-Detecting WOM-Code**.

The definition of the decoding map in the second case is extended to be  $\mathcal{D}_{C_W} : \{0, 1\}^n \rightarrow \{0, 1\}^k \cup \{\mathbf{F}\}$ , where the symbol  $\mathbf{F}$  indicates an error detection flag.

**Remark 2.** If after decoding on the  $i$ -th write, a cell which is in physical state zero is found to contain an error, this error can be corrected (at least theoretically) prior to the next write by changing the physical state of this cell to a one. However, if after decoding on the  $i$ -th write, a cell which is in physical state one is found to be in error, the physical state of this cell cannot be changed prior to the next write. In this case, however, it is assumed that the encoder on the  $(i+1)$ -st write knows that the cell's true state is a zero. There is no problem if the encoder wants to write a one in this cell. However, if the encoder wants to write a zero in this cell, the error which was corrected on the  $i$ -th write will also occur on the  $(i+1)$ -st write. When we say that a WOM-code is an  $e$ -error-correcting code we mean that the code will correct  $e$  or a fewer errors on each write but we realize that errors which were corrected on one write could appear on subsequent writes. This information could be used in decoding but the decoder we consider here does not do so.

In this work, we present error-detecting and error-correcting WOM-codes, which have the following generic scheme:

- 1) We assume that there exists a  $\mathcal{WC}[n, k, t]$  WOM-code  $C_W(\mathcal{E}_{C_W}, \mathcal{D}_{C_W})$ . Its  $n$  cells are denoted by  $c = (c_0, \dots, c_{n-1})$  and called the **information cells**. Note that this original code  $C_W$  cannot correct errors.
- 2) Our constructed code consists of the  $n$  information cells  $c$ , and  $r$  more cells, called the **redundancy cells**,  $p = (p_0, \dots, p_{r-1})$ . The redundancy cells enable the decoder to correct cell-errors. That is, we get a  $\mathcal{WC}[n+r, k, t]$  WOM-code with some error correction capabilities.

### III. SINGLE-ERROR-DETECTING WOM-CODES

In this Section we present single-error-detecting WOM-codes. As described in Section II,  $C_W(\mathcal{E}_{C_W}, \mathcal{D}_{C_W})$  is a  $\mathcal{WC}[n, k, t]$  WOM-code, and its cells, called the information cells, are  $c = (c_0, \dots, c_{n-1})$ . We construct a  $\mathcal{WC}[n+t, k, t]$  single-error-detecting WOM-code,  $C_{SED}(\mathcal{E}_{C_{SED}}, \mathcal{D}_{C_{SED}})$ .

In this construction there are  $t$  redundancy cells, denoted by  $p = (p_0, p_1, \dots, p_{t-1})$ , i.e., the value of  $r$  in the generic

scheme is  $t$ . The code  $C_{SED}$  satisfies the following property: at each write, the parity of the  $t$  redundancy cells,  $\sum_{i=0}^{t-1} p_i$ , and the parity of the  $n$  information cells,  $\sum_{i=0}^{n-1} c_i$ , are the same.

In the encoding map  $\mathcal{E}_{C_{SED}}$ , the new data vector  $v$  is encoded in the  $n$  information cells by  $\mathcal{E}_{C_W}(c, v)$ . If the parity of the information cells is changed, then one of the  $t$  redundancy cells is programmed. Since there are initially  $t$  redundancy cells in state zero and each time at most one of them is programmed, there is at least one unprogrammed cell at each write.

In the decoding map  $\mathcal{D}_{C_{SED}}$ , at most one of the cells is in error. If the information cell's parity is different than the redundancy cell's parity, then the flag  $\mathbf{F}$  is returned to indicate a single error. Otherwise, the data vector  $v$  is simply decoded by  $v = \mathcal{D}_{C_W}(c)$ . Thus, we have proved the following Theorem.

**Theorem 1.** If  $C_W$  is a  $\mathcal{WC}[n, k, t]$  WOM-code, then  $C_{SED}$  is a  $\mathcal{WC}[n+t, k, t]$  single-error-detecting WOM-code.

This scheme can be applied to all known WOM-codes. In the next example, we show how to adapt it to WOM-codes which are based on Hamming codes.

**Example 1.** In [2], a construction of WOM-codes, based on Hamming codes, is presented. The construction provides a  $\mathcal{WC}[2^k - 1, k, 2^{k-2} + 2]$  WOM-code for  $k \geq 4$ , and a  $\mathcal{WC}[2^k - 1, k, 2^{k-2} + 1]$  WOM-code for  $k = 2, 3$ . In particular, the  $\mathcal{WC}[3, 2, 2]$  WOM-code, presented by Rivest and Shamir [13], is a special case of this construction for  $k = 2$ . Later, in [7] the case  $k \geq 4$  was improved and  $\mathcal{WC}[2^k - 1, k, 5 \cdot 2^{k-4} + 1]$  WOM-codes were presented. For  $k \geq 4$ , it is possible to show that besides the first write, the number of programmed cells at each write is even [15]. Therefore, the parity bit changes its values at most one. Thus, one redundancy cell is sufficient and we get a  $\mathcal{WC}[2^k, k, 5 \cdot 2^{k-4} + 1]$  single-error-detecting code. In fact, a similar construction to this code was presented in [15]. For  $k = 2, 3$ , the construction is slightly modified. At each write, the redundancy cells parity and the information cells parity are opposite. Then, we can show that at most  $t-1$  cells are sufficient and thus a  $\mathcal{WC}[2^k + t - 1, k, 2^{k-2} + 2]$  single-error-detecting code exists. The following table demonstrates the construction for the  $\mathcal{WC}[3, 2, 2]$  WOM-code. The bit in bold represents the redundancy cell.

Bits Value	First Write	Second Write
00	000 <b>1</b>	11 <b>10</b>
01	001 <b>0</b>	11 <b>01</b>
10	010 <b>0</b>	10 <b>11</b>
11	100 <b>0</b>	01 <b>11</b>

### IV. SINGLE-ERROR-CORRECTING WOM-CODES

As described in Section III,  $C_W(\mathcal{E}_{C_W}, \mathcal{D}_{C_W})$  is a  $\mathcal{WC}[n, k, t]$  WOM-code, and its information cells are  $c = (c_0, \dots, c_{n-1})$ . We add  $r$  redundancy cells,  $p = (p_0, \dots, p_{r-1})$ , where we assume that  $C_{WD}(\mathcal{E}_{C_{WD}}, \mathcal{D}_{C_{WD}})$  is a  $\mathcal{WC}[r, \lceil \log_2(n+1) \rceil, t]$  single-error-detecting WOM-code. We construct a  $\mathcal{WC}[n+r, k, t]$  single-error-correcting WOM-code,  $C_{SEC}$ .

At each write we generate a  $\lceil \log_2(n+1) \rceil$ -bit vector, called the **syndrome** and denoted by  $s$ . The syndrome will correspond to the redundancy bits of a Hamming code (or a shortened Hamming code) of length  $n$ , and will enable us to locate an information cell in error.

Next, and in the following sections, we provide the exact specification of the encoding and decoding maps, which are

described algorithmically by a pseudo-code notation. We let  $\alpha$  be a primitive element in the extension field  $GF(2^{\lceil \log_2(n+1) \rceil})$ .

**Encoding map  $\mathcal{E}_{C_{SEC}}$ :** The input is the memory-state vector  $(c, p)$ , and the new  $k$ -bit data vector  $v$ . The output is either a new memory-state vector  $(c', p')$  or the erasure symbol E.

```

1.  $c' = \mathcal{E}_{C_W}(c, v)$ ;
2. if ( $c' == E$ ) return E;
3.  $s = \sum_{i=0}^{n-1} c'_i \alpha^i$ ;
4.  $p' = \mathcal{E}_{C_{WD}}(p, s)$ ;
5. if ( $p' == E$ ) return E;
6. return  $(c', p')$ ;

```

In the encoding map  $\mathcal{E}_{C_{SEC}}$ , first, the data vector  $v$  is encoded in the information cells  $c$  (line 1). If writing does not succeed, the symbol is returned E (line 2). Otherwise, the syndrome  $s$  of the new  $n$  information cells is calculated (line 3). Then,  $s$  is encoded in the redundancy cells using the map  $\mathcal{E}_{C_{WD}}(p, s)$  (line 4). If this writing fails, the symbol E is returned (line 5); if not, the new memory-state vector is returned (line 6).

**Decoding map  $\mathcal{D}_{C_{SEC}}$ :** The input is the memory-state vector  $(c', p')$ . The output is the decoded  $k$ -bit data vector  $v$ .

```

1.  $s'' = \mathcal{D}_{C_{WD}}(p')$ ;
2. if ( $s'' == F$ )
3.    $\{ v = \mathcal{D}_{C_W}(c'); \text{ return } v; \}$ 
4.  $s' = \sum_{i=0}^{n-1} c'_i \alpha^i$ ;
5. if ( $s' == s''$ )
6.    $\{ v = \mathcal{D}_{C_W}(c'); \text{ return } v; \}$ 
7.  $i = \log_{\alpha}(s' + s'')$ ;
8.  $v = \mathcal{D}_{C_W}(c'_0, \dots, c'_{i-1}, 1 - c'_i, c'_{i+1}, \dots, c'_{n-1})$ ;
9. return  $v$ ;

```

The syndrome  $s''$  is decoded from the redundancy cells  $p'$  (line 1). The code  $C_{WD}$  is a single-error-detecting WOM-code and hence by its decoding map  $\mathcal{D}_{C_{WD}}$  it is possible to know if there is an error in one of the  $r$  redundancy cells (line 2).

- 1) If there is a redundancy cell error (the condition in line 2 holds), then there is no error in the information cells and  $v$  is decoded by the decoding map  $\mathcal{D}_{C_W}$  (line 3).
- 2) If there is no error in the redundancy cells, then  $s''$  is the correct value of the syndrome  $s$ . The received syndrome  $s'$  from the received  $n$  cells is  $s' = \sum_{i=0}^{n-1} c'_i \alpha^i$  (line 4). If  $s' = s''$  (line 5), then there is no error in the  $n$  information cells. Otherwise, if the  $i$ -th cell is in error then  $s' + s'' = \alpha^i$ . We assume that the operator  $\log_{\alpha}(s' + s'')$  returns the value  $i$  such that  $\alpha^i = s' + s''$  (line 7). Therefore, we know the correct value of the information cells and can decode the data vector.

**Theorem 2.** If  $C_W$  is a  $\mathcal{WC}[n, k, t]$  WOM-code,  $C_{WD}$  is a  $\mathcal{WC}[r, \lceil \log_2(n+1) \rceil, t]$  single-error-detecting WOM-code, then  $C_{SEC}$  is a  $\mathcal{WC}[n+r, k, t]$  single-error-correcting WOM-code.

**Example 2.** As in Example 1, for the code  $C_W$ , we use the  $\mathcal{WC}[2^k - 1, k, 5 \cdot 2^{k-4} + 1]$  WOM-codes for  $k \geq 4$  from [7]. Therefore,  $n = 2^k - 1$ , and  $\lceil \log_2(n+1) \rceil = k$ , so we can use the  $\mathcal{WC}[2^k, k, 5 \cdot 2^{k-4} + 1]$  single-error-detecting WOM-code also from Example 1. The generated  $\mathcal{WC}[2 \cdot 2^k - 1, k, 5 \cdot 2^{k-4} + 1]$  single-error-correcting WOM-code has WOM-rate

$$\mathcal{R}^t = \frac{k(5 \cdot 2^{k-4} + 1)}{2 \cdot 2^k - 1} \geq \frac{\log_2(3.2(t-1))}{6.4},$$

which is an improvement upon the constructions in [14] and the trivial construction we presented in the Introduction.

## V. DOUBLE-ERROR-CORRECTING WOM-CODES

The double-error-correcting WOM-codes construction is very similar to the single-error-correcting case in Section IV. Here, we show how to modify the latter construction and present the encoding and decoding maps. The same WOM-codes  $C_W, C_{WD}$  are used. There are  $2r$  redundancy cells, divided into two  $r$ -cell groups,  $p_1 = (p_0, p_1, \dots, p_{r-1})$  and  $p_2 = (p_r, p_{r+1}, \dots, p_{2r-1})$ . The redundancy cells group  $p_1, p_2$  stores a  $\lceil \log_2(n+1) \rceil$ -bit syndrome vector  $s_1, s_2$ , respectively. The two syndromes correspond to the two roots  $\alpha, \alpha^3$  of the double-error-correcting BCH code, denoted by  $C_{2-BCH}$ , where  $\alpha$  is a primitive element in the field  $GF(2^{\lceil \log_2(n+1) \rceil})$ . In this construction  $\lceil \log_2(n+1) \rceil$  is assumed to be an odd integer. The code is denoted by  $C_{DEC}$ .

For the decoding map  $\mathcal{D}_{C_{DEC}}$ , we use the single-error-correcting WOM-code decoding map  $\mathcal{D}_{C_{SEC}}$ , which receives as its input  $n$  information cells and  $r$  redundancy cells. Note that while the code  $C_{SEC}$  uses a fixed primitive element  $\alpha \in GF(2^{\lceil \log_2(n+1) \rceil})$ , it is possible to use any other primitive element in this field. We slightly modify the input arguments of the decoding map  $\mathcal{D}_{C_{SEC}}$  such that the primitive element is its first parameter. The modified decoding map is denoted by  $\mathcal{D}'_{C_{SEC}}$ . We use the decoding map  $\mathcal{D}_{C_{2-BCH}}$  of the double-error-correcting BCH code. Its input is the  $2 \lceil \log_2(n+1) \rceil$  syndrome bits; its output is the error vector.

Due to the lack of space, we only show the encoding and decoding maps and leave out their detailed explanation.

**Encoding map  $\mathcal{E}_{C_{DEC}}$ :** The input is the memory-state vector  $(c, p_1, p_2)$  and the  $k$ -bit data vector  $v$ . The output is a new memory-state vector  $(c', p'_1, p'_2)$  or the erasure symbol E.

```

1.  $c' = \mathcal{E}_{C_W}(c, v)$ ;
2. if ( $c' == E$ ) return E;
3.  $s_1 = \sum_{i=0}^{n-1} c'_i \alpha^i$ ;  $s_2 = \sum_{i=0}^{n-1} c'_i \alpha^{3i}$ ;
4.  $p'_1 = \mathcal{E}_{C_{WD}}(p_1, s_1)$ ;  $p'_2 = \mathcal{E}_{C_{WD}}(p_2, s_2)$ ;
5. if ( $(p'_1 == E) \text{ OR } (p'_2 == E)$ ) return E;
6. return  $(c', p'_1, p'_2)$ ;

```

**Decoding map  $\mathcal{D}_{C_{DEC}}$ :** The input is the memory-state vector  $(c', p'_1, p'_2)$ . The output is the decoded  $k$ -bit data vector  $v$ .

```

1.  $s''_1 = \mathcal{D}_{C_{WD}}(p'_1)$ ;  $s''_2 = \mathcal{D}_{C_{WD}}(p'_2)$ ;
2. if ( $s''_1 == F$ )
3.    $\{ v = \mathcal{D}'_{C_{SEC}}(\alpha^3, c', p'_2); \text{ return } v; \}$ 
4. if ( $s''_2 == F$ )
5.    $\{ v = \mathcal{D}'_{C_{SEC}}(\alpha, c', p'_1); \text{ return } v; \}$ 
6.  $s'_1 = \sum_{i=0}^{n-1} c'_i \alpha^i$ ;  $s'_2 = \sum_{i=0}^{n-1} c'_i \alpha^{3i}$ ;
7. if ( $(s'_1 == s''_1) \text{ OR } (s'_2 == s''_2)$ )
8.    $\{ v = \mathcal{D}_{C_W}(c'); \text{ return } v; \}$ 
9.  $e' = \mathcal{D}_{C_{2-BCH}}(s'_1 + s''_1, s'_2 + s''_2)$ ;
10.  $v = \mathcal{D}_{C_W}(c' + e')$ ;
11. return  $v$ ;

```

**Theorem 3.** If  $C_W$  is a  $\mathcal{WC}[n, k, t]$  WOM-code,  $C_{WD}$  is a  $\mathcal{WC}[r, \lceil \log_2(n+1) \rceil, t]$  single-error-detecting WOM-code, and  $\lceil \log_2(n+1) \rceil$  is an odd integer, then  $C_{DEC}$  is a  $\mathcal{WC}[n+2r, k, t]$  double-error-correcting WOM-code.

The construction does not work if  $\lceil \log_2(n+1) \rceil$  is even since  $\alpha^3$  is no longer primitive, and thus the decoding map in line 3 cannot succeed. It is possible to modify this construction in case  $\lceil \log_2(n+1) \rceil$  is even by adding  $t$  more cells.

## VI. TRIPLE-ERROR CORRECTING WOM-CODES

From the previous sections we deduce that a generic scheme to construct an  $e$ -error correcting WOM-code is to combine an existing WOM-code and an  $e$ -error-correcting code, where the latter code is defined by  $e$  roots  $\alpha_1, \dots, \alpha_e$ . However, not every  $e$ -error-correcting code can suit this scheme. For example, in the double-error-correcting construction in Section V, the BCH code with roots  $\alpha$  and  $\alpha^3$  cannot work if  $\lceil \log_2(n+1) \rceil$  is even. The construction does not work since  $\alpha^3$  is not a primitive element and hence the code generated only by  $\alpha^3$  is not a single-error-correcting code. For arbitrary  $e$ , if the cyclic  $e$ -error-correcting code is defined by  $e$  roots, then a necessary but not sufficient condition for this scheme to work is that every subset of  $k \leq e$  roots generates a cyclic  $k$ -error-correcting code. We state this property in the following Definition.

**Definition.** Let  $n$  be an integer and  $\alpha_1, \dots, \alpha_e$  be  $e$  different elements in the field  $GF(2^n)$ . Let the code  $\mathcal{C}(\alpha_1, \dots, \alpha_e)$  be a cyclic error-correcting code of length  $2^n - 1$  with roots  $\alpha_1, \dots, \alpha_e$ . The code  $\mathcal{C}(\alpha_1, \dots, \alpha_e)$  is called a **strong  $e$ -error-correcting code** if for every  $1 \leq k \leq e$  and every set of  $k$  different elements  $\alpha_{i_1}, \dots, \alpha_{i_k} \in \{\alpha_1, \dots, \alpha_e\}$ , the code  $\mathcal{C}(\alpha_{i_1}, \dots, \alpha_{i_k})$  is a  $k$ -error-correcting code.

We note that finding strong  $e$ -error-correcting codes is a fascinating problem by itself but is beyond the scope of this paper. Next, we show how to choose the roots  $\alpha_1, \alpha_2, \alpha_3$  such that  $\mathcal{C}(\alpha_1, \alpha_2, \alpha_3)$  is a strong triple-error-correcting code. For the following discussion,  $\alpha$  is assumed to be a primitive element in  $GF(2^n)$ . The following result was proved in [11].

**Theorem 4.** Let  $n$  be an odd integer and  $\gcd(k, n) = 1$ . Then,  $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a triple-error-correcting code.

In [1], the authors show an alternative proof to the last Theorem and state the following Lemma.

**Lemma 5.** Let  $n$  be an integer and  $\gcd(k, n) = 1$ . Then,  $\mathcal{C}(\alpha, \alpha^{2^k+1})$  is a double-error-correcting code.

We state the following Lemma, but skip its proof.

**Lemma 6.** Let  $n$  be an integer such that  $\gcd(n, 6) = 1$ , and let  $k = \frac{n-1}{2}$ . Then, the following properties hold.

- 1)  $\mathcal{C}(\alpha), \mathcal{C}(\alpha^{2^k+1}), \mathcal{C}(\alpha^{2^{3k}+1})$  are single-error-correcting codes.
- 2)  $\mathcal{C}(\alpha, \alpha^{2^k+1}), \mathcal{C}(\alpha, \alpha^{2^{3k}+1})$  are double-error-correcting codes.
- 3)  $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a triple-error-correcting code.

In order to prove that  $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a strong triple-error-correcting code we must show that  $\mathcal{C}(\alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a double-error-correcting code. Before proving the next Lemma, we state the definition of an almost perfect nonlinear mapping.

**Definition.** A mapping  $f : GF(p^n) \rightarrow GF(p^n)$  is called an **almost perfect nonlinear (APN) mapping** if each equation

$$f(x+a) - f(x) = b$$

for  $a, b \in GF(p^n)$  and  $a \neq 0$  has at most two solutions in  $GF(p^n)$ . If  $f$  is an APN mapping and is of the form  $f(x) = x^d$  then  $f$  is called an **almost perfect nonlinear power mapping**.

The next Lemma was proved in [11].

**Lemma 7.** If  $n$  is an odd integer,  $2 \leq k \leq \frac{n-1}{2}$ , and  $\gcd(n, k) = 1$  then the mapping  $f(x) = x^{2^{2k}-2^k+1}$  over  $GF(2^n)$  is an APN mapping.

The proof of the next Lemma follows an outline similar to that of the proof of Theorem 1 in [1].

**Lemma 8.** If  $n, k$  are integers,  $\gcd(n, 6) = 1$  and  $k = \frac{n-1}{2}$ , then  $\mathcal{C}(\alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a double-error-correcting code.

*Proof:* Note first that  $\alpha^{2^k+1}$  is a primitive element in  $GF(2^n)$ . Since  $\gcd(n, k) = 1$  and  $n$  is odd, according to Lemma 7,  $f(x) = x^d$  is an APN power mapping, where  $d = 2^{2k} - 2^k + 1$ . We denote  $\gamma = \alpha^{2^k+1}$ , and hence need to prove that  $\mathcal{C}(\gamma, \gamma^d)$  is a double-error-correcting code.

Assume to the contrary that the code is not a double-error-correcting code. Then, there exist four integers  $0 \leq i_1 < i_2 < i_3 < i_4 \leq 2^n - 2$  such that

$$\begin{aligned} \gamma^{i_1} + \gamma^{i_2} + \gamma^{i_3} + \gamma^{i_4} &= 0 \\ (\gamma^{i_1})^d + (\gamma^{i_2})^d + (\gamma^{i_3})^d + (\gamma^{i_4})^d &= 0. \end{aligned}$$

The last two equations can be written also as follows

$$\begin{aligned} \gamma^{i_1} + \gamma^{i_2} &= a = \gamma^{i_3} + \gamma^{i_4} \\ (\gamma^{i_1})^d + (\gamma^{i_2})^d &= b = (\gamma^{i_3})^d + (\gamma^{i_4})^d, \end{aligned}$$

for some  $a, b \in GF(2^n)$ , and  $a \neq 0$ . Hence, the equation  $(x+a)^d + x^d = b$  has four different solutions:  $\gamma^{i_1}, \gamma^{i_2}, \gamma^{i_3}, \gamma^{i_4}$ . This is a contradiction since  $x^d$  is an APN mapping. ■

**Theorem 9.** If  $n, k$  are integers,  $\gcd(n, 6) = 1$ , and  $k = \frac{n-1}{2}$ , then  $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$  is a strong triple-error-correcting code.

We now show the triple-error-correcting WOM-code construction. Again, we use the WOM-codes  $\mathcal{C}_W, \mathcal{C}_{WD}$ , and assume that  $\gcd(\lceil \log_2(n+1) \rceil, 6) = 1$  and  $\alpha$  is primitive in  $GF(2^{\lceil \log_2(n+1) \rceil})$ . The strong triple-error-correcting is denoted by  $\mathcal{C}_3^{\text{strong}}(\mathcal{E}_{\mathcal{C}_3^{\text{strong}}}, \mathcal{D}_{\mathcal{C}_3^{\text{strong}}})$ . Its roots are  $\alpha_1 = \alpha, \alpha_2 = \alpha^{2^k+1}, \alpha_3 = \alpha^{2^{3k}+1}$ , where  $k = \frac{\lceil \log_2(n+1) \rceil - 1}{2}$ . There are  $3r + t$  redundancy cells, divided into four groups:

- 1) The first  $t$  cells  $q = (q_0, \dots, q_{t-1})$  are used with the  $n$  information cells to construct a  $\mathcal{WC}[n+t, k, t]$  single-error-detecting WOM-code  $\mathcal{C}'_W(\mathcal{E}_{\mathcal{C}'_W}, \mathcal{D}_{\mathcal{C}'_W})$ .
- 2) Three groups of  $r$  cells each:  $p_1 = (p_{0, \dots, p_{r-1}})$ ,  $p_2 = (p_{r, \dots, p_{2r-1}})$ , and  $p_3 = (p_{2r, \dots, p_{3r-1}})$ . The  $i$ -th group,  $i = 1, 2, 3$ , stores the  $\lceil \log_2(n+1) \rceil$ -bit syndrome  $s_i$  which corresponds to the root  $\alpha_i$ .

To conclude, we construct a  $\mathcal{WC}[n+t+3r, k, t]$  triple-error-correcting WOM-code,  $\mathcal{C}_{\text{TEC}}(\mathcal{E}_{\mathcal{C}_{\text{TEC}}}, \mathcal{D}_{\mathcal{C}_{\text{TEC}}})$ .

**Encoding map  $\mathcal{E}_{\mathcal{C}_{\text{TEC}}}$ :** The input is the memory-state  $(c, q, p_1, p_2, p_3)$  and the data vector  $v$ . The output is a new memory-state  $(c', q', p'_1, p'_2, p'_3)$  or the erasure symbol E.

1.  $(c', q') = \mathcal{E}_{\mathcal{C}'_W}((c, q), v)$ ;
2. **if**  $((c', q') == E)$  **return** E;
3.  $s_1 = \sum_{i=0}^{n-1} c_i \alpha_1^{id_1}$ ;  $s_2 = \sum_{i=0}^{n-1} c_i \alpha_2^{id_2}$ ;  $s_3 = \sum_{i=0}^{n-1} c_i \alpha_3^{id_3}$ ;
4.  $p'_1 = \mathcal{E}_{\mathcal{C}_{WD}}(p_1, s_1)$ ;  $p'_2 = \mathcal{E}_{\mathcal{C}_{WD}}(p_2, s_2)$ ;
5.  $p'_3 = \mathcal{E}_{\mathcal{C}_{WD}}(p_3, s_3)$ ;
6. **if**  $((p'_1 == E) \text{ OR } (p'_2 == E) \text{ OR } (p'_3 == E))$  **return** E;
7. **return**  $(c', q', p'_1, p'_2, p'_3)$ ;

In the decoding map, we use the decoding map  $\mathcal{D}_{\mathcal{C}_{\text{TEC}}}$ . Note that in  $\mathcal{D}_{\mathcal{C}_{\text{TEC}}}$ , instead of using a double-error-correcting BCH code, we can use any other cyclic double-error-correcting code which is given by its two roots. Line 9 of the decoding map  $\mathcal{D}_{\mathcal{C}_{\text{TEC}}}$  is modified with the decoding map of the new double-error-correcting code. The input to the modified decoding map

$\mathcal{D}'_{C_{DEC}}$  is the two roots of the cyclic double-error-correcting code, the  $n$  information cells, and the  $2r$  redundancy cells, corresponding to the two syndromes of the roots.

**Decoding map  $\mathcal{D}_{C_{TEC}}$ :** The input is the memory state  $(c', q', p'_1, p'_2, p'_3)$ . The output is the decoded data vector  $v$ .

```

1.  $s''_1 = \mathcal{D}_{C_{WD}}(p'_1)$ ;  $s''_2 = \mathcal{D}_{C_{WD}}(p'_2)$ ;  $s''_3 = \mathcal{D}_{C_{WD}}(p'_3)$ ;
2. if ( $s''_1 == \mathbf{F}$ )
3.   {  $v = \mathcal{D}'_{C_{DEC}}(\alpha_2, \alpha_3, c', p'_2, p'_3)$ ; return  $v$ ; }
4. if ( $s''_2 == \mathbf{F}$ )
5.   {  $v = \mathcal{D}'_{C_{DEC}}(\alpha_1, \alpha_3, c', p'_1, p'_3)$ ; return  $v$ ; }
6. if ( $s''_3 == \mathbf{F}$ )
7.   {  $v = \mathcal{D}'_{C_{DEC}}(\alpha_1, \alpha_2, c', p'_1, p'_2)$ ; return  $v$ ; }
8.  $s'_1 = \sum_{i=0}^{n-1} c'_i \alpha_1^i$ ;  $s'_2 = \sum_{i=0}^{n-1} c'_i \alpha_2^i$ ;  $s'_3 = \sum_{i=0}^{n-1} c'_i \alpha_3^i$ ;
9.  $e_1 = s'_1 + s''_1$ ;  $e_2 = s'_2 + s''_2$ ;  $e_3 = s'_3 + s''_3$ ;
10. if ( $(\sum_{i=0}^{n-1} c'_i) == (\sum_{i=0}^{t-1} q'_i)$ )
11.   {  $v = \mathcal{D}'_{C_{DEC}}(\alpha_1, \alpha_2, c', p'_1, p'_2)$ ; return  $v$ ; }
12. if ( $(e_1^{2^k+1} == e_2)$  OR ( $e_1^{2^{3k+1}} == e_3$ )
      OR ( $e_2^{2^k-2^{k+1}} == e_3$ ))
13.   {  $v = \text{maj}(\mathcal{D}'_{C_{SEC}}(\alpha_1, c', p'_1), \mathcal{D}'_{C_{SEC}}(\alpha_2, c', p'_2),$ 
       $\mathcal{D}'_{C_{SEC}}(\alpha_3, c', p'_3))$ ; return  $v$ ; }
14.  $e' = \mathcal{D}_{C_{strong}}(e_1, e_2, e_3)$ ;
15.  $v = \mathcal{D}_{C_{W}}(c' + e')$ ;
16. return  $v$ ;

```

First, the three syndromes from the redundancy cell groups are decoded (line 1). If the decoded syndrome  $s''_1$  gets the error flag  $\mathbf{F}$  (line 2), then there is at least one error in the group  $p'_1$ . In the information cells  $c'$  and  $p'_2, p'_3$  there are at most two errors. Therefore, we decode by applying the map  $\mathcal{D}'_{C_{DEC}}$  on  $c'$  and  $p'_2, p'_3$  with the roots  $\alpha_1, \alpha_3$  (line 3). The same procedure is applied if  $s''_2$  or  $s''_3$  gets the error flag  $\mathbf{F}$  (lines 4 – 7). Here, we use the property of  $\mathcal{C}_3^{\text{strong}}$  that every two out of its three roots generate a double-error-correcting code.

In line 8, all the syndromes  $s''_1, s''_2, s''_3$  do not get the error flag  $\mathbf{F}$ . The received syndromes  $s'_1, s'_2, s'_3$  from the received cells and the differences  $e_1, e_2, e_3$  are calculated in lines 8, 9. If the condition in line 10 holds, then the cells  $c'$  and  $q'$  have zero or two errors. In both cases, the cells  $c', p'_1, p'_2$  have at most two errors so it is possible to decode as done in line 11.

We are left with the case where the parities of the cells  $c'$  and  $q'$  are not the same. That is, these cells have either one or three errors. We prove this case in the next Lemma.

**Lemma 10.** *The condition in line 12 holds if and only if there is a single error in the cells  $c'$  and  $q'$ .*

*Proof:* If the cells  $c'$  and  $q'$  have one error, then there is at most one error in  $c'$ . At most one of the redundancy cell groups  $p'_1, p'_2, p'_3$  has two errors, that is, at least two of these groups do not have errors. If there is no error in the first and second groups and the  $i$ -th information cell  $c'_i$  is in error, then  $e_1 = \alpha_1^i = \alpha^i$  and  $e_2 = \alpha_2^i = \alpha^{i(2^k+1)}$ . Therefore,  $e_1^{2^k+1} = e_2$ . This condition clearly holds also if there are no errors in the cells  $c'$ . Similarly, if there is no error in  $p'_1$  and  $p'_3$  then  $e_1^{2^{3k+1}} = e_3$ , and if there is no error in  $p'_2$  and  $p'_3$  then  $e_2^{2^k-2^{k+1}} = e_3$ . Therefore, if there is a single error in the cells  $c'$  and  $q'$  then the condition in line 12 holds.

Now we show that if there is not a single error in the cells  $c'$  and  $q'$  then the condition in line 12 does not hold. In this

case, the cells  $c'$  and  $q'$  have three errors and there is no error in the redundancy cells  $p'_1, p'_2, p'_3$ . Assume that the information cells have three errors in locations  $i, j, \ell$ . Then,

$$\begin{aligned} e_1 &= \alpha^i + \alpha^j + \alpha^\ell \\ e_2 &= \alpha^{i(2^k+1)} + \alpha^{j(2^k+1)} + \alpha^{\ell(2^k+1)} \\ e_3 &= \alpha^{i(2^{3k+1})} + \alpha^{j(2^{3k+1})} + \alpha^{\ell(2^{3k+1})} \end{aligned}$$

for some  $0 \leq i < j < \ell \leq n-1$ . In this case,  $e_1^{2^k+1} \neq e_2$ . Otherwise, we get

$$\begin{aligned} e_1 + \alpha^i + \alpha^j + \alpha^\ell &= 0 \\ e_1^{(2^k+1)} + \alpha^{i(2^k+1)} + \alpha^{j(2^k+1)} + \alpha^{\ell(2^k+1)} &= 0, \end{aligned}$$

and  $\mathcal{C}(\alpha, \alpha^{2^k+1})$  has a codeword of weight at most four, which is a contradiction. Similarly,  $e_1^{2^{3k+1}} \neq e_3$  and  $e_2^{2^k-2^{k+1}} \neq e_3$ . The case of two errors in the information cells is the same. Hence, the condition in line 12 does not hold. ■

According to Lemma 10, if the condition in line 12 holds then there is at most one error in the information cells. At most one of the redundancy cell groups  $p'_1, p'_2, p'_3$  has errors. Therefore, at least two out of the three decoding maps in line 13 succeed, and the function **maj**, which outputs the majority of the three decoded values, returns the correct value of  $v$ . In line 14, there are at most three errors in the information cells and no errors in the cells  $p'_1, p'_2, p'_3$ , so it is possible to find the error vector (line 14) and decode (line 15). We conclude with the following Theorem.

**Theorem 11.** *If  $\mathcal{C}_W$  is a  $\mathcal{WC}[n, k, t]$  WOM-code,  $\mathcal{C}_{WD}$  is a  $\mathcal{WC}[r, \lceil \log_2(n+1) \rceil, t]$  single-error-detecting WOM-code, and  $\text{gcd}(\lceil \log_2(n+1) \rceil, 6) = 1$ , then  $\mathcal{C}_{TEC}$  is a  $\mathcal{WC}[n+3r+t, k, t]$  triple-error-correcting WOM-code.*

## REFERENCES

- [1] C. Bracken and T. Helleseht, "Triple-error-correcting BCH-like codes," *Proc. IEEE International Symposium on Inform. Theory*, pp. 1723–1725, Seoul, Korea, June 2009.
- [2] G.D. Cohen, P. Godlewski, and F. Merx, "Linear binary code for write-once memories," *IEEE Trans. Inform. Theory*, vol. 32, no. 5, pp. 697–700, September 1986.
- [3] H. Dobbertin, "Almost perfect nonlinear power functions on  $\text{GF}(2^n)$ : the Welch case," *IEEE Trans. Inform. Theory*, vol. 45, no. 4, pp. 1271–1275, May 1999.
- [4] F. Fu and A.J. Han Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph," *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 308–313, September 1999.
- [5] A. Fiat and A. Shamir, "Generalized write-once memories," *IEEE Trans. Inform. Theory*, vol. 30, pp. 470–480, September 1984.
- [6] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, vol. 37, pp. 138–163, June 2005.
- [7] P. Godlewski, "WOM-codes construits à partir des codes de Hamming," *Discrete Math.*, no. 65 pp. 237–243, 1987.
- [8] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inform. Theory*, vol. 30, pp. 470–480, September 1984.
- [9] A. Jiang, "On the generalization of error-correcting WOM codes," *Proc. IEEE International Symposium on Inform. Theory*, pp. 1391–1395, Nice, France, June 2007.
- [10] A. Jiang and J. Bruck, "Joint coding for flash memory storage," *Proc. IEEE International Symposium on Inform. Theory*, pp. 1741–1745, Toronto, Canada, July 2008.
- [11] T. Kasami, "The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes," *Information and Control*, vol. 18, pp. 369–394, September 1971.
- [12] H. Mahdavi, P.H. Siegel, A. Vardy, J.K. Wolf, and E. Yaakobi, "A nearly optimal construction of flash codes," *Proc. IEEE International Symposium on Inform. Theory*, pp. 1239–1243, Seoul, Korea, July 2009.
- [13] R.L. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and Control*, vol. 55, nos. 1–3, pp. 1–19, December 1982.
- [14] G. Zémor and G.D. Cohen, "Error-correcting WOM-codes," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pp. 730–734, May 1991.
- [15] G. Zémor, "Problèmes combinatoires liés à l'écriture sur des mémoires," Ph.D. Dissertation, ENST, Paris, France, November 1989.